Client-Driven Content Extraction Associated with Table

K.C. Santosh and Abdel Belaïd
LORIA - Université de Lorraine
54506 Vandoeuvre-lès- Nancy, France
{santosh.kc, abdel.belaid}@loria.fr

Abstract

The goal of the project is to extract content within table in document images based on learnt patterns. Real-world users i.e., clients first provide a set of key fields within the table which they think are important. These are first used to represent the graph where nodes are labelled with semantics including other features and edges are attributed with relations. Attributed relational graph (ARG) is then employed to mine similar graphs from a document image. Each mined graph will represent an item within the table, and hence a set of such graphs will compose a table. We have validated the concept by using a real-world industrial problem.

1 Introduction

In document analysis and processing, table extraction from document images has been received an important attention since it contains key information. In the context of table extraction [1–4], document image analysis and processing basically describes table either in terms of lines and (un)analysed text blocks, a set of cells resembling the two-dimensional grid or a set of strings that are integrated with each other via relations, for instance.

Basically, table detection and its structure recognition are two major tasks. Table detection can be taken as a primary issue, which is however does not provide a complete solution [5] since one needs to be able to extract key fields within it. Existing methods such as table segmentation [6] do not extract key fields, nor do they explicitly perform the content understanding [7]. Note that structural information by considering relations between the contents, for instance can be very useful in indexing and retrieving document information [2]. To analyse table-forms structure, rulings techniques are basically limited without a priori knowledge about table organisation [1]. Such concepts are completely failed since not all tables possess graphical lines. Besides, plain ascii texts, text blocks are used. Detecting columns, lines and headers, and representing them in terms of graph, for instance is interesting since it contains structural information. In order to fully exploit table in the scanned documents rather than just outlining the overall boundary, it is interesting to extract those fields that are important or meaningful for the clients. To handle this, in this paper, key fields are provided by the clients. These key fields are then used to build a graph so that it can be applied for table extraction in the absence of clients.

The rest of the paper is organised as follows. We start with explaining the proposed method in Section 2. Full experiments are reported and analysed in Section 3. The paper is concluded in Section 4.

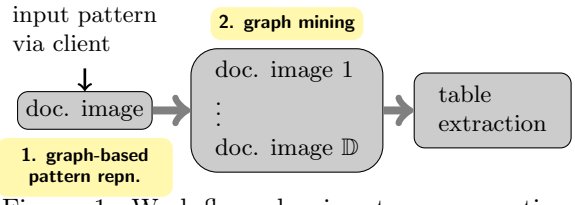


Figure 1. Work-flow showing two consecutive phases: graph-based pattern representation and graph mining, to handle table extraction.

2 Proposed method

Generally speaking, table is composed of similar items (sometimes just a single) even when columns alignment and corresponding text flow (either in a single or multiple lines) are not guaranteed. Given an input pattern (i.e., an item, for instance) from a client, finding similar patterns from the document is the core part of the paper. It not only extracts important fields (in accordance with the client) but also configures table represented by a set of similar patterns. To handle this, we first represent an input pattern via an ARG and perform graph mining so that similar graphs can be extracted that are structurally and semantically similar. Fig. 1 shows a screen-shot of the overall idea.

2.1 Graph-based pattern representation

In any document d, the clients provide input pattern(s) while showing the interest of the particular type t of table in either header, body or footer zone: $\text{table}_t = \{\text{pattern}_n, n \in [1, \mathbb{N}]\}$, where $\mathbb{N}$ can be arbitrary. An example of input pattern is shown in Fig. 2 i.e., it is just a collection of the selected key fields: $\{\text{field}_i\}_{i=1}^{\mathbb{A}}$. To represent each field, we define a feature set $\mathcal{F}$ as $\{\text{feature}_f\}_{f=1}^{\mathbb{F}}$. For any i-th field, we can formally represent feature as $\text{field}_i^{\mathcal{F}} = \{$

(`box`: [left, top, right, bottom]);	(`wSep`: words separation);	
(`value`: content);	(`noW`: number of words);	
(`type`: content type);	(`noL`: number of lines);	(1)
(`size`: string length);	(`label`: *date* and *price*, for instance.)$\}$	

The labels are the derivative of features, representing semantic values via regular expressions. Thanks to the regular expressions, we are able to express a wide range of string values even when we have possible OCR errors due to broken characters and characters are connected with graphics, for instance. To exploit relative positioning between the key fields, we basically use bounding box and its projection into 3×3 partitions [8] (defined in IR^2 i.e., *left*, *right*, ...). For more precision, we integrate the level of neighbourhood k into the basic predefined set of spatial predicates, we have

$$r_{ij} = \textit{spatial predicate}_{k_1,k_2}(\text{field}_i, \text{field}_j). \quad (2)$$

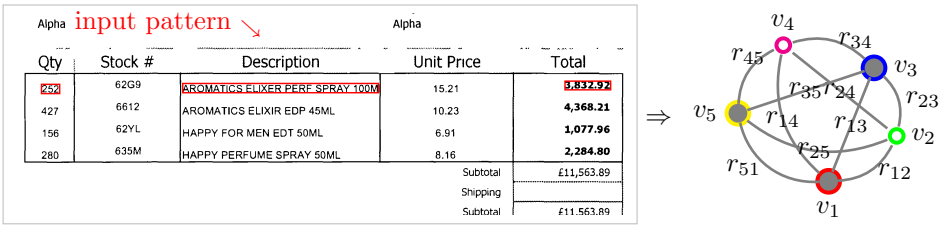


Figure 2. An example of the input pattern and the corresponding graph that includes missing fields.

Formally, $k = 0$ for an adjacent (an immediate field), and k varies from 1 to $\mathbb{A} - 1$ for non-adjacent ones. Note that k_1 and k_2 represent horizontal and vertical orientations, respectively.

Now, we introduce a 4-tuple ARG

$$G(V, E, F_V, F_E),$$

where

- V is a finite set of nodes (fields);
- $E \subseteq V \times V$ i.e., a finite set of edges and each $r_{ij} \in E$ is a pair of (v_i, v_j) where $v_i, v_j \in V$;
- $F_V : V \rightarrow L_V$, L_V represents a set of nodes as well as their labels $\mathcal{L}$; and
- $F_E : E \rightarrow R_E$, R_E represents the edges via relations.

To make graph complete, we also include non-selected fields which are mainly missing and neighbouring fields. To know how many words can be taken for a single field, we simply use intra-field (i.e., maximum distance between the words in a single field) knowledge from the selected key fields.

2.2 Content extraction via graph mining

Given the pattern graph Q, to extract similar graphs from a document, it starts with pivotal nodes selection in a document and perform relation assignment to compute feature score between the pairs of nodes. Relations assignment repeats until a similar graph G is achieved, with respect to Q.

Pivotal nodes selection. In a predefined set $\mathcal{L}$ of labels such as *price*, *date*, *address* and *description* in the domain, for every node v_i^q in pattern graph Q, the corresponding label $\ell_i^q \in \mathcal{L}$ is defined i.e., $V^q = \{(v_i^q, \ell_i^q), i = 1 \ldots \mathbb{V}^q\}$. Having these labelled nodes $\{(v_i^q, \ell_i^q)\}$ in a pattern graph Q, the target is to select nodes sharing identical labels $\{(v_i, \ell_i)\}$ from a document d. We now, refer the selected nodes as pivotal nodes.

Feature score computation. Each pivotal node is taken and started to validate relations with neighbouring nodes in a document, as in pattern graph. To compute feature score between the pair of nodes (v_i, v_j) in a document with respect to $(v_i^q, v_j^q) \in Q$, their respective relations must be identical i.e., r_{ij}^q validates with r_{ij}. More formally, we can compute feature score between two corresponding nodes v^q and v as $f.\text{score}(v^q, v) =$

$$\begin{cases} 1: \text{ label in } v^q = \text{ label in } v, \text{ and} \\ \frac{1}{\mathbb{F}} \sum_f \lambda_f \times s_{v^q, v}^{\text{feature}_f} : \text{ otherwise,} \end{cases} \quad (3)$$

where $\lambda_f \in [0, 1]$ provides weight to each features used to compute feature matching score $s_{(,)}$. For each particular feature, weight λ_f can be varied according to its robustness and so is application dependent. Given two strings: x reference and y primary, we compute feature (like string *value*, number of *word*s and *size* (*cf.* Eq. (1))) matching scores as follows.

• String *type*:
$s_{x,y}^{type} = 1 - \big(\text{Levenshtein dist.}(x, y)/\max(x, y)\big)$, where we treat numerals $\{0-9\}$, all alphabets $\{A-Z, a-z\}$ and symbols equally.

• Number of *word*s in a string:
$s_{x,y}^{word} = 1 - \big(\text{dist.}^{word}(x, y)/\max(x, y)\big)$ i.e., an absolute difference in number words is normalised by the maximum number of words.

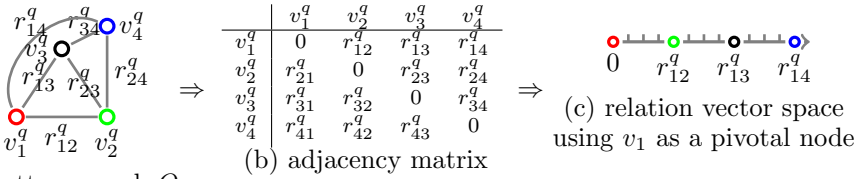
• String *size*:
$s_{x,y}^{length} = 1 - \big(\text{dist.}^{length}(x, y)/\max(x, y)\big)$ i.e., an absolute difference in size (number of letters) is normalised by its maximum size.

Following Fig. 3, let us elaborate a concept of matching. To simplify the explanation, let us first create a relation vector space from a pattern graph and then realise the assignment process for each pivotal node in a document. Taking a single pivotal node v_1 from a data graph G (having identical label with respect to v_1^q in Q i.e., $\ell_1^p = \ell_1^q \in \mathcal{L}$), the idea is to assign relations $\{r_{12}^q, r_{13}^q, r_{14}^q\}$ in data graph G. We validate relations $\{r_{12}, r_{13}\}$ one-by-one and compute feature score in parallel. It provides $G \subseteq Q$. However, an addition of a node v_3 can help to make them exactly similar in configuration via an edit cost operation.

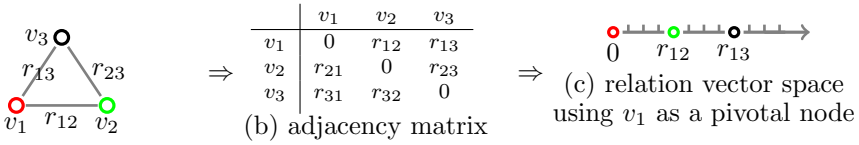
Graph matching score computation. An aggregation of both scores i.e., $r.\text{score}$ from relation assignment and $f.\text{score}$ from feature computation between the nodes yields a matching score S for data graph G with respect to Q

$$S(Q, G) = \alpha \frac{1}{\mathbb{R}} \sum_{i,j \in \mathcal{R}^q, i \neq j} r.\text{score}(r_{i,j}^q, r_{i,j}) + \quad (4)$$
$$(1 - \alpha) \frac{1}{\mathbb{V}^q} \sum_{i \in V^q} f.\text{score}(v_i^q, v_i), \alpha \in [0, 1].$$

Confidence score computation. From each input pattern, a set of mined graphs $\{(G_g, S_g)\}$ will represent a table i.e., an output. For such an output, we compute corresponding confidence score (CS). CS is computed from the aggregation of all matching scores $\{S_g\}_{g=1}^{\mathbb{G}}$, which is then normalised i.e., $\text{CS}_{\mathbb{k}}^{t_n} = \frac{1}{\mathbb{G}} \sum_{g=1}^{\mathbb{G}} S_g$. In case of multiple input patterns, the outputs are ranked and provided on a one-to-one basis. Ranking is based on the order of similarity.



(a) pattern graph Q (b) adjacency matrix (c) relation vector space using v_1 as a pivotal node



(a) data graph G (b) adjacency matrix (c) relation vector space using v_1 as a pivotal node

Figure 3. Relation vector space to simplify relation assignment. In this illustration, it shows two different graphs: Q and G, the corresponding adjacency matrices and relation vector spaces for a single pivotal node v_1.

Note that we aim to use set of mined graphs to iteratively update the pattern graph and transform into a graph model so that it can be used in the absence of the clients – which is beyond the scope of the paper. A proof of the concept is reported in [9] and the thorough extension (aiming to apply document information content extraction, not necessarily be always found in structured documents like forms) has been made in [10].

3 Experiments

3.1 Dataset and evaluation metric

Dataset. We work on a real-world industrial problem in direct collaboration with the **ITESOFT**[1], France. Currently, the dataset is composed of 15 classes with 100 samples per class. For each document, clients provide ground-truths i.e., all similar patterns within the table, according to the pattern selected.

Evaluation metric. An output i.e., the detected table is represented by a collection of mined graphs $O = \{G_g, S_g\}$ in a test document, and there are $\mathbb{G}^\circ$ list of ground-truthed patterns corresponding to the ground-truthed table $O^\circ = \{G_g^\circ\}_{g^\circ=1}^{\mathbb{G}^\circ}$. Each graph G has a number of fields that are simply represented by iconic boxes $\{B_b\}_{b=1}^{\mathbb{B}}$.

To evaluate, we extend the area-ratio-based measure proposed by Shafait and Smith [11]. It uses bounding boxes to describe detected tables and the ground-truths. In our framework, the overlapping ratio between the two boxes is defined as $OR_1(B_b^\circ, B_b) = \frac{2\times|B_b^\circ \cap B_b|}{|B_b^\circ|+|B_b|}$, where $|B_b^\circ \cap B_b|$ is the intersected or common area of two bounding boxes from ground-truthed and detected table respectively and $|B_b^\circ|, |B_b|$ are the individual areas. Note that $OR_1(,) \in [0,1]$. We sum up all $OR_1(,)$ and normalise to compute overall overlapping ratio between ground-truth pattern G° and detected pattern G by $OR_2(G^\circ, G) = \frac{1}{\max(\mathbb{B}^\circ,\mathbb{B})} \sum OR_1(B_b^\circ, B_b), \{b^\circ : b^\circ \in \mathbb{B}^\circ \wedge b \in \mathbb{B}^\circ\}$. Then for a whole table, we can express evaluation metric as

$$Eval(O^\circ, O) = \frac{1}{\max(\mathbb{G}^\circ,\mathbb{G})} \sum OR_2(G_g^\circ, G_g), \quad \{g^\circ : g^\circ \in O^\circ \wedge g \in O^\circ\}. \tag{5}$$

[1] http://www.itesoft.com.

3.2 Results and analysis

We have validated the outputs over 15 different suppliers by taking the associated ground-truths and reported the average performance in Table 1. More specifically, it provides the two different ways to evaluate:

1. one is associated with the input pattern created in the laboratory and
2. another one is directly related with client or real-world patterns.

The first evaluation of course, aims to provide an overall concept that can be applied to content extraction associated with the table. The latter one provides how robust it is. In the reported results in Table 1, we observe the following.

1. Without a surprise, cleaner the input pattern, better the performance. This happens to be in *eval. 1* since input patterns are created in accordance with what OCR results.
2. In contrast, in case of the client input patterns (*eval. 2*), a single field selection may sometimes take word(s) from another closer fields (can be left or right), and multiple lines. In that selected box (from clients), since OCR reads some dots (due to noise) as 'full-stop', 'colon' and 'semi-colon', it does not allow possible cleaning. As a consequence, feature properties representing the graph nodes can possibly varied. Fig. 6. shows an example of it.

Besides, another considerable issue is the complexity of the graph-based pattern representation. In case of input patterns with complex structural formats (lets say zig-zag), such non-selected fields integration makes pattern graph more complex. Furthermore, as said before, our system performance has been affected due to OCR errors since the system does not provide the

Table 1. Average performance (in %) over three different types of table: header, body and footer.

Table type⇒	Header	Body	Footer	Avg.
Eval. 1	97	99	98	98
Eval. 2	96	98	95	97

Eval. 1: input patterns created in lab.
Eval. 2: input patterns from clients.
Execution time $\simeq$ 2 sec./doc. image.

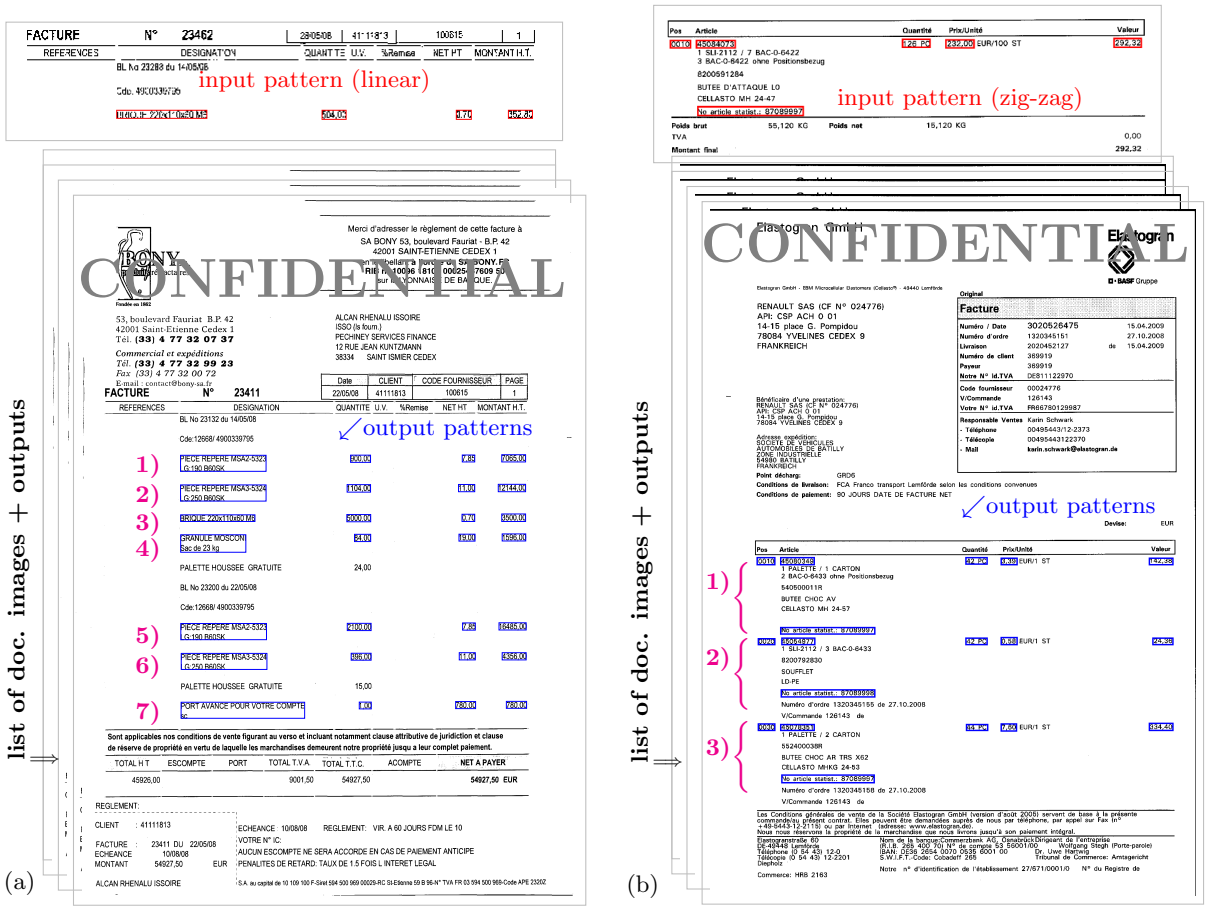


Figure 4. Examples showing content extraction within the table in accordance with the input pattern (from client). Tables are composed of separately (a) seven and (b) three similar patterns in two different suppliers.

expected semantics label at nodes in the graph. An example of the OCR effect is 'false detection' because of the structural similarity between the graphs.

4 Conclusions and future perspectives

In this paper, we have presented client-driven pattern-based approach to table extraction via graph mining scheme, inspiring from a real-world applications. We have very much focused and validated that the table extraction does not always mean only to detect the presence and absence as well as to spot the area where table(s) is(are) located but also to select important key fields within it while rejecting others.

Given an input pattern (i.e., a pattern graph), finding similar pattern graphs so that we can reinforce or update it iteratively each time we extract them, is one of the primary issues of the further work [9, 10], for instance. As a consequence, such models are used to exploit document information content in the absence of clients.

References

[1] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *IJDAR*, 7(1):1–16, 2004.

[2] B. Coüasnon, "Dmos, a generic document recognition method: application to table structure analysis in a general and in a specific way," *IJDAR*, 8(2-3):111–122, 2006.

[3] M. Hurst, "Towards a theory of tables," *IJDAR*, 8(2-3): 123–131, 2006.

[4] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy, "Table-processing paradigms: a research survey," *IJDAR*, 8(2-3): 66–86, 2006.

[5] S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda, "A simple and effective table detection system from document images," *IJDAR*, 8(2-3): 172–182, 2006.

[6] Y. Liang, Y. Wang, and E. Saund, "A method of evaluating table segmentation results based on a table image ground truther," in *Proceedings of ICDAR*, 2011, pp. 247–251.

[7] F. Deckert, B. Seidler, M. Ebbecke, and M. Gillmann, "Table content understanding in smartfix," in *Proceedings of ICDAR*, 2011, pp. 488–492.

[8] D. Papadias and Y. Theodoridis, "Spatial relations, minimum bounding rectangles, and spatial data structures," *IJGIS*, 11(2): 111–138, 1997.

[9] K. C. Santosh and A. Belaïd, "Pattern-based approach to table extraction," in *Proceedings of IbPRIA*. Springer, 2013, to appear.

[10] ——, "Document information extraction and its evaluation based on client's relevance," in *Proceedings of ICDAR*, 2013, submitted.

[11] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *Proceedings of DAS*, 2010, pp. 65–72.